ns-2 Tutorial

Polly Huang
USC/ISI
huang@isi.edu
http://www-scf.usc.edu/~bhuang
14 June, 1999

Essentials & Getting Started

Outlines

- **Essentials**
- Getting Started
- Fundamental tcl, otcl and ns
- Case Studies
  - Web, TCP, Routing, Queuing

Object-Oriented

+ Reusability
+ Maintainability
– Careful Planning Ahead

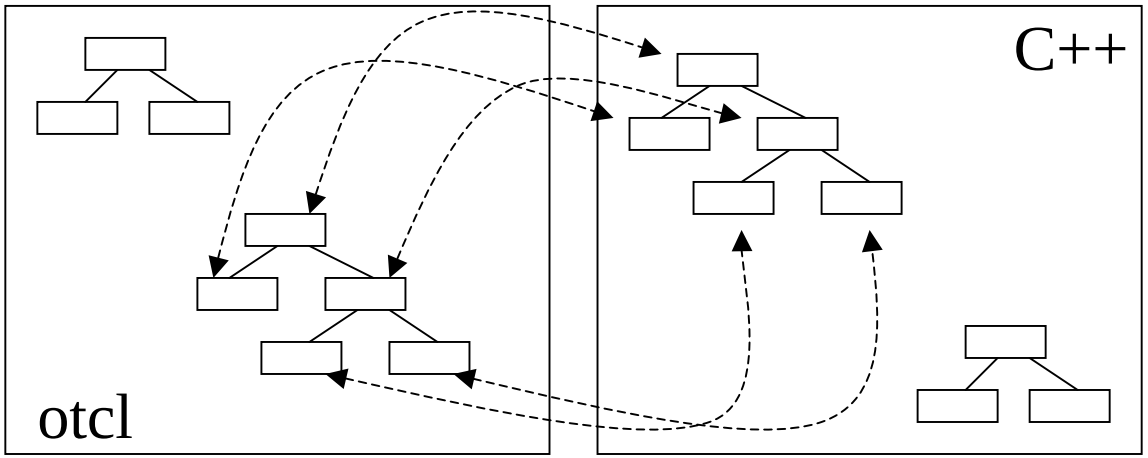
C++ and otcl Separation

- C++ for data
  - per packet action
- otcl for control
  - periodic or triggered action

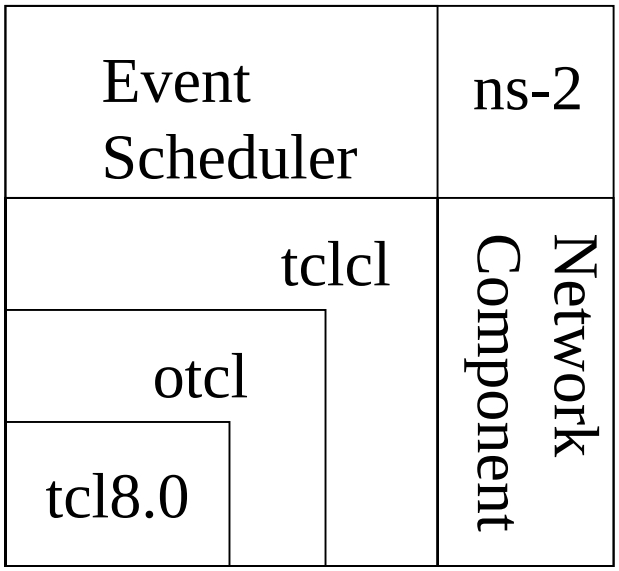
+ Compromize between composibility and speed

– Learning & debugging

otcl and C++: The Duality



tcl Interpreter With Extents



- otcl: Object-oriented support
- tclcl: C++ and otcl linkage
- Discrete event scheduler
- Data network (the Internet) components

Outlines

- Essentials
- **Getting Started**
- Fundamental tcl, otcl and ns
- Case Studies
  - Web, TCP, Routing, Queuing

Installation

- Getting the pieces
  - tcl/tk8.0, otcl, tclcl, ns-2, (and nam-1)
- http://www-mash.cs.berkeley.edu/ns/ns-build.html
- ns-users@mash.cs.berkeley.edu
  - majordomo@mash.cs.berkeley.edu
  - subscribe ns-users yourname@address

Hello World - Interactive Mode

swallow 71% **ns**

% set ns [new Simulator]

_o3

% $ns at 1 “puts \“Hello World!\””

1

% $ns at 1.5 “exit”

2

% $ns run

Hello World!

swallow 72%

Hello World - Passive Mode

simple.tcl

```
set ns [new Simulator]
$ns at 1 “puts \“Hello World!\””
$ns at 1.5 “exit”
$ns run
```

swallow 74% **ns simple.tcl**

Hello World!

swallow 75%

Outlines

- Essentials
- Getting Started
- **Fundamental tcl, otcl and ns**
- Case Studies
  - Web, TCP, Routing, Queuing, Wireless

Fundamentals

- tcl
- otcl:
  - ftp://ftp.tns.lcs.mit.edu/pub/otcl/doc/tutorial.html
- ns-2
  - http://www-mash.cs.berkeley.edu/ns/nsDoc.ps.gz
  - http://www-mash.cs.berkeley.edu/ns/ns-man.html

Basic tcl

```
proc test {} {
    set a 43
    set b 27
    set c [expr $a + $b]
    set d [expr [expr $a - $b] * $c]
    for {set k 0} {$k < 10} {incr k} {
        if {$k < 5} {
            puts “k < 5, pow= [expr pow($d, $k)]”
        } else {
            puts “k >= 5, mod= [expr $d % $k]”
        }
    }
}
test
```

Basic otcl

```
Class mom
mom instproc greet {} {
    $self instvar age_
    puts "$age_ years old mom: How are you doing?"
}

Class kid -superclass mom
kid instproc greet {} {
    $self instvar age_
    puts "$age_ years old kid: What's up, dude?"
}
```

```
set a [new mom]
$a set age_ 45
set b [new kid]
$b set age_ 15

$a greet
$b greet
```

Basic ns-2

- Creating the event scheduler
- Creating network
- Computing routes
- Creating connection
- Creating traffic
- Inserting errors
- Tracing

Creating Event Scheduler

- Create scheduler
  - set ns [new Simulator]
- Schedule event
  - $ns at <time> <event>
  - <event>: any legitimate ns/tcl commands
- Start scheduler
  - $ns run

Creating Network

- Nodes
  - set n0 [$ns node]
  - set n1 [$ns node]
- Links & Queuing
  - $ns duplex-link $n0 $n1 <bandwidth> <delay> <queue_type>
  - <queue_type>: DropTail, RED, CBQ, FQ, SFQ, DRR

Creating Network: LAN

- LAN
  - $ns make-lan <node_list> <bandwidth> <delay> <ll_type> <ifq_type> <mac_type> <channel_type>
  - <ll_type>: LL
  - <ifq_type>: Queue/DropTail,
  - <mac_type>: MAC/802_3
  - <channel_type>: Channel

Computing routes

- Unicast
  - $ns rtproto <type>
  - <type>: Static, Session, DV, cost, multi-path
- Multicast
  - $ns multicast (right after [new Simulator])
  - $ns mrtproto <type>
  - <type>: CtrMcast, DM, ST, BST

Creating Connection: UDP

- UDP
  - set udp [new Agent/UDP]
  - set null [new Agent/NULL]
  - $ns attach-agent $n0 $udp
  - $ns attach-agent $n1 $null
  - $ns connect $udp $null

Creating Connection: TCP

- TCP
  - set tcp [new Agent/TCP]
  - set tcpsink [new Agent/TCPSink]
  - $ns attach-agent $n0 $tcp
  - $ns attach-agent $n1 $tcpsink
  - $ns connect $tcp $tcpsink

Creating Traffic: On Top of TCP

- FTP
  - set ftp [new Application/FTP]
  - $ftp attach-agent $tcp
- Telnet
  - set telnet [new Application/Telnet]
  - $telnet attach-agent $tcp

Creating Traffic: On Top of UDP

- CBR
  - set src [new Application/Traffic/CBR]
- Exponential or Pareto on-off
  - set src [new Application/Traffic/Exponential]
  - set src [new Application/Traffic/Pareto]

Creating Traffic: Trace Driven

- Trace driven
  - set tfile [new Tracefile]
  - $tfile filename <file>
  - set src [new Application/Traffic/Trace]
  - $src attach-tracefile $tfile
- <file>:
  - Binary format
  - inter-packet time (msec) and packet size (byte)

Inserting Errors

- Creating Error Module
  - set loss_module [new ErrorModel]
  - $loss_module set rate_ 0.01
  - $loss_module unit pkt
  - $loss_module ranvar [new RandomVariable/Uniform]
  - $loss_module drop-target [new Agent/Null]
- Inserting Error Module
  - $ns lossmodel $loss_module $n0 $n1

Tracing

- Trace packets on all links
  - $ns trace-all [open test.out w]

```
<event> <time> <from> <to> <pkt> <size>--<flowid> <src> <dst> <seqno> <aseqno>
     + 1 0 2 cbr 210 ------- 0 0.0 3.1 0 0
     - 1 0 2 cbr 210 ------- 0 0.0 3.1 0 0
     r 1.00234 0 2 cbr 210 ------- 0 0.0 3.1 0 0
```

- Trace packets on all links in nam-1 format
  - $ns namtrace-all [open test.nam w]

Outlines

- Essentials
- Getting Started
- Fundamental tcl, otcl and ns-2
- **Case Studies**

Case Studies

- Routing - Multicast (mcast.tcl)
- TCP (tcp.tcl)
- Web (web.tcl & dumbbell.tcl)
- Queuing - RED (red.tcl)

Visualization Tools

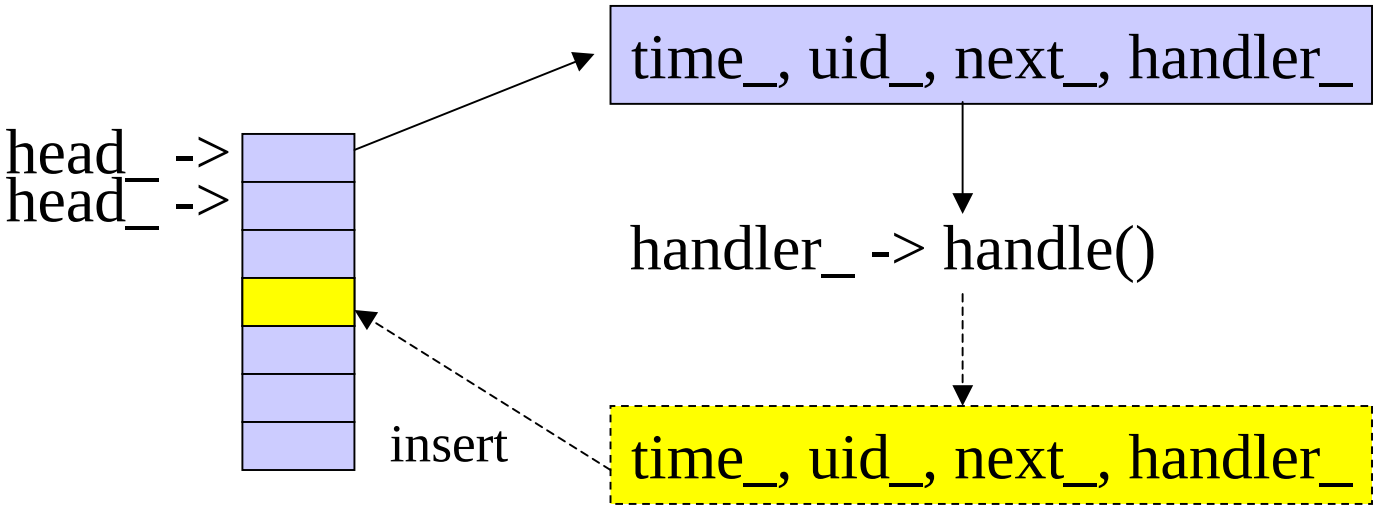
- nam-1 (Network AniMator Version 1)
- xgraph

ns-2 Internal

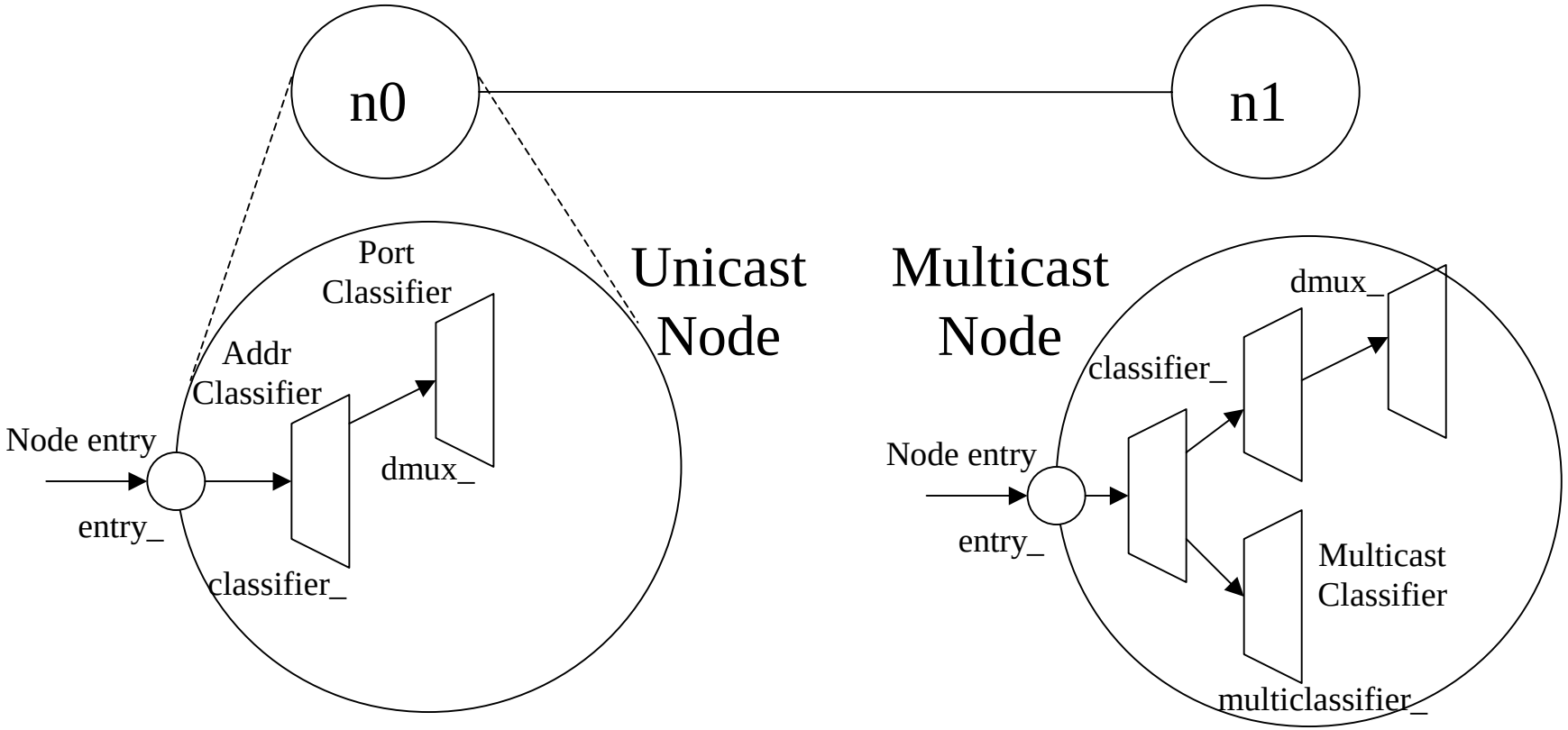
Internals

- Discrete Event Scheduler
- Network Topology
- Routing
- Transport
- Application
- Packet Flow
- Packet Format

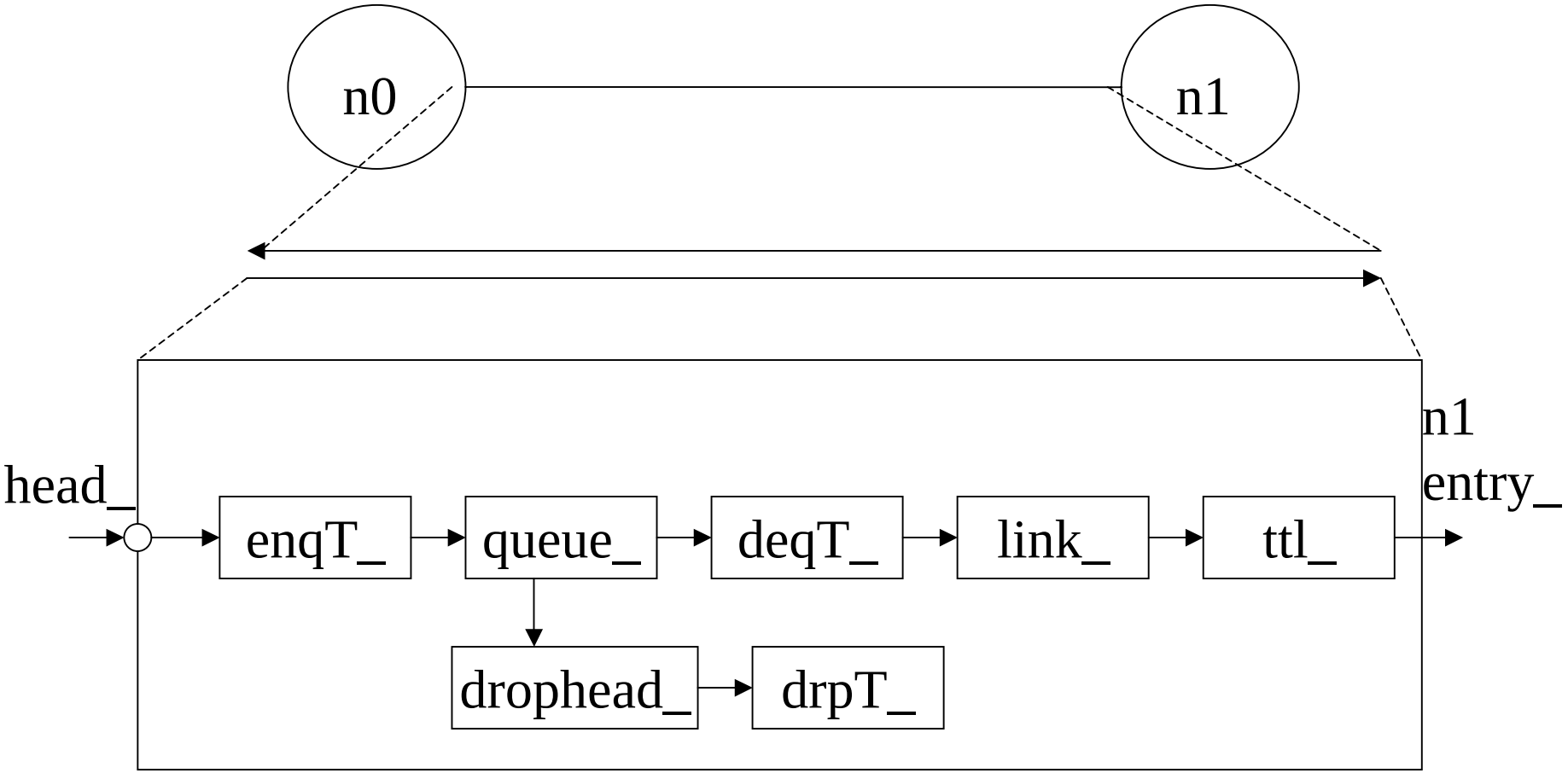
Discrete Event Scheduler



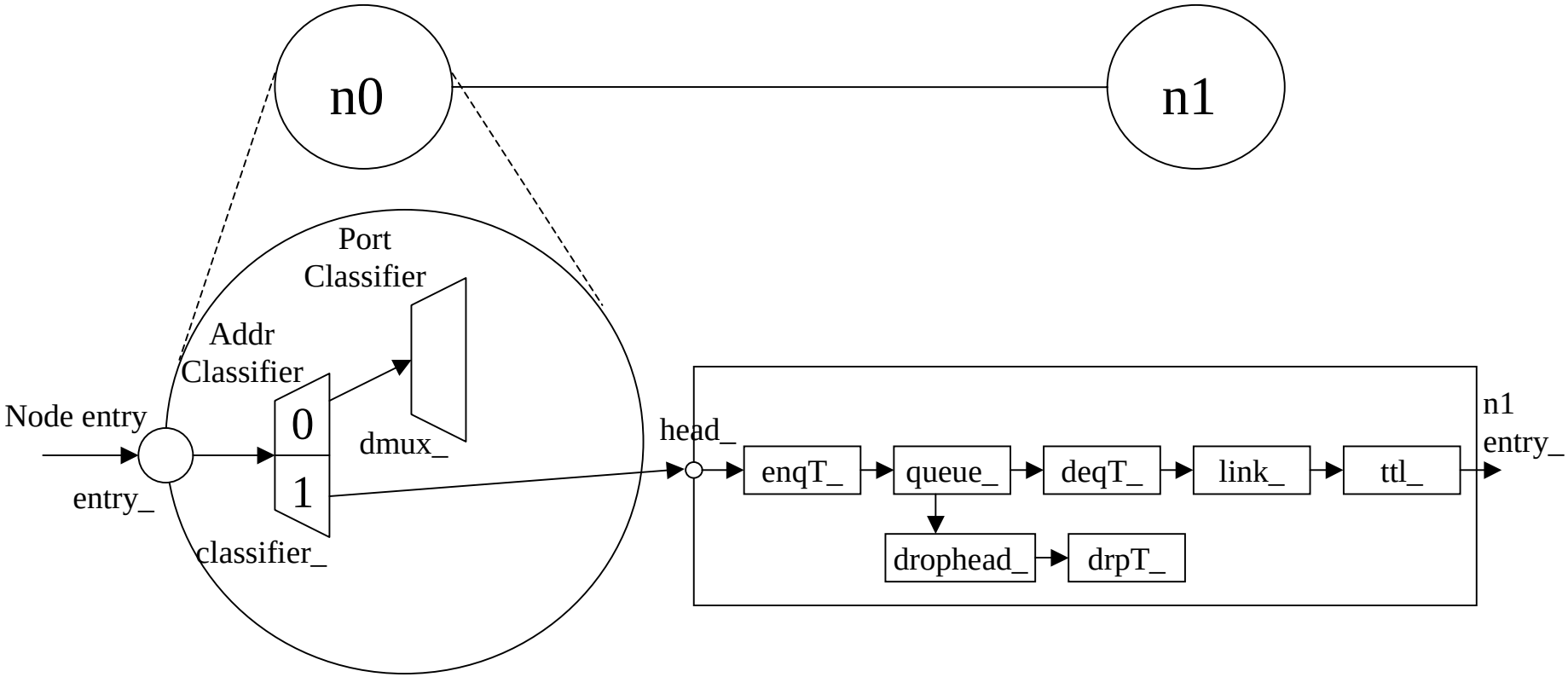
Network Topology - Node



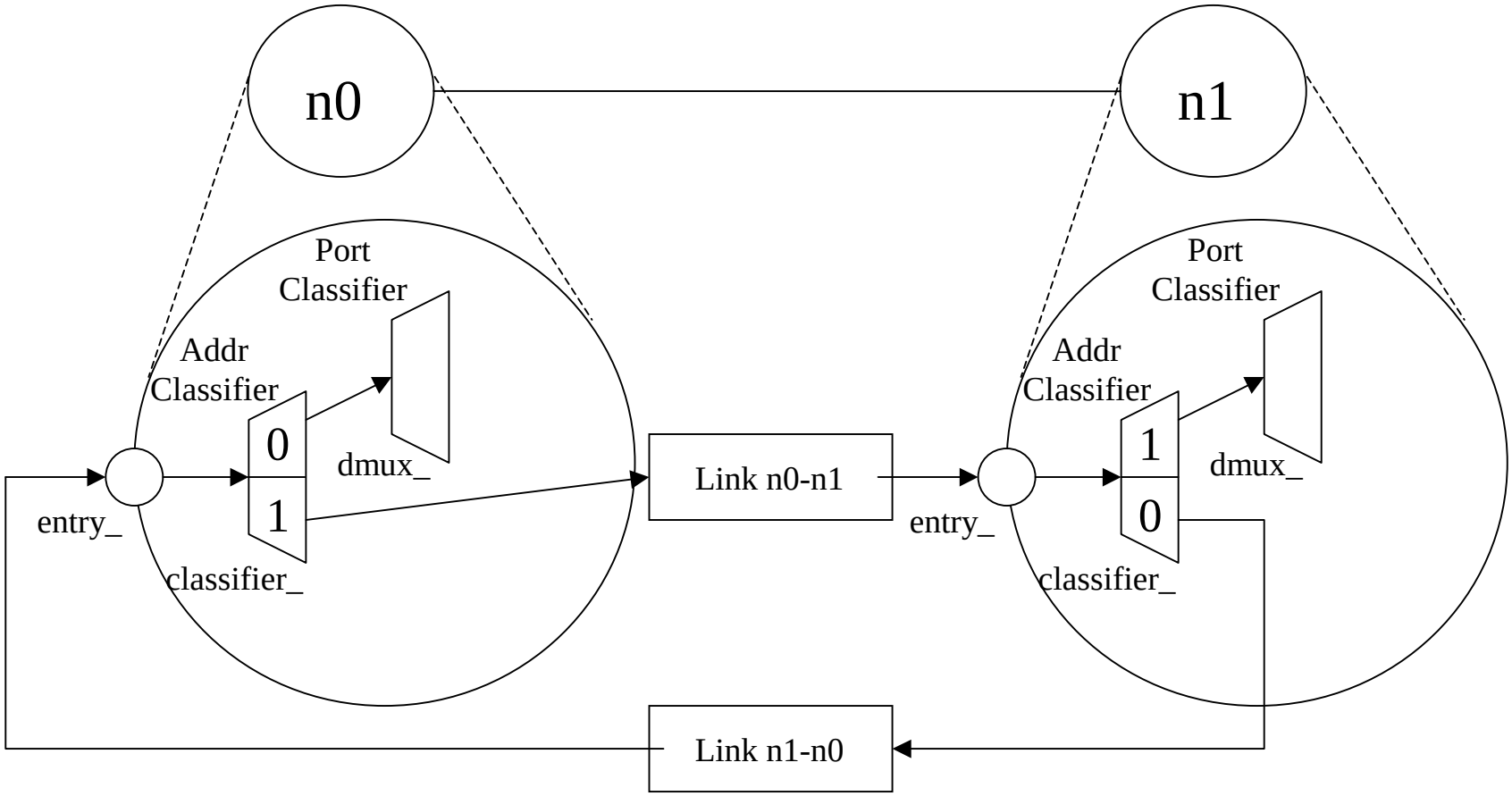
Network Topology - Link



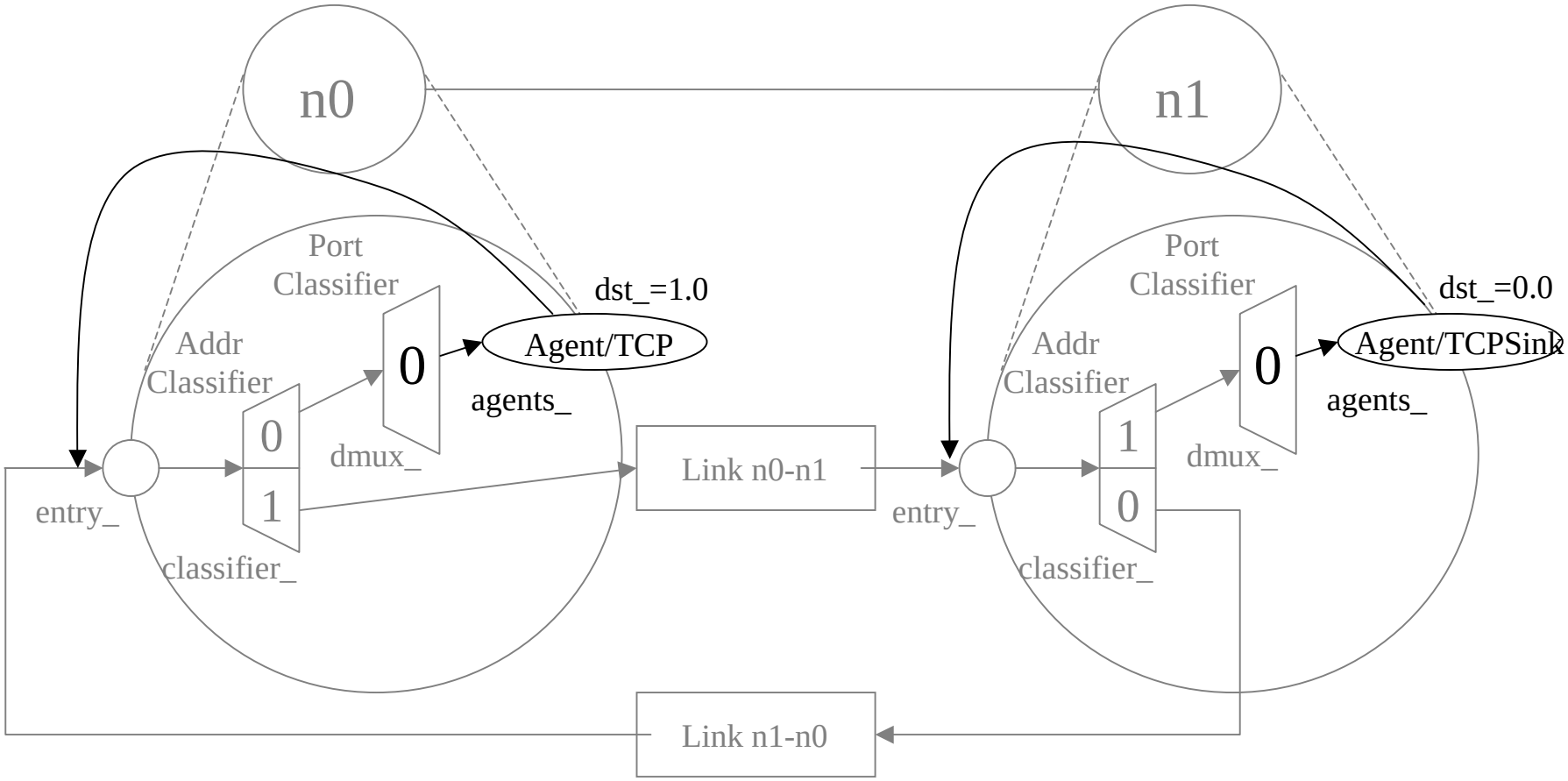
Routing


n0
n1
Port
Classifier
Addr
Classifier
Node entry
0
1
dmux_
entry_
classifier_
head_
enqT_
queue_
deqT_
link_
ttl_
n1
entry_
drophead_
drpT_

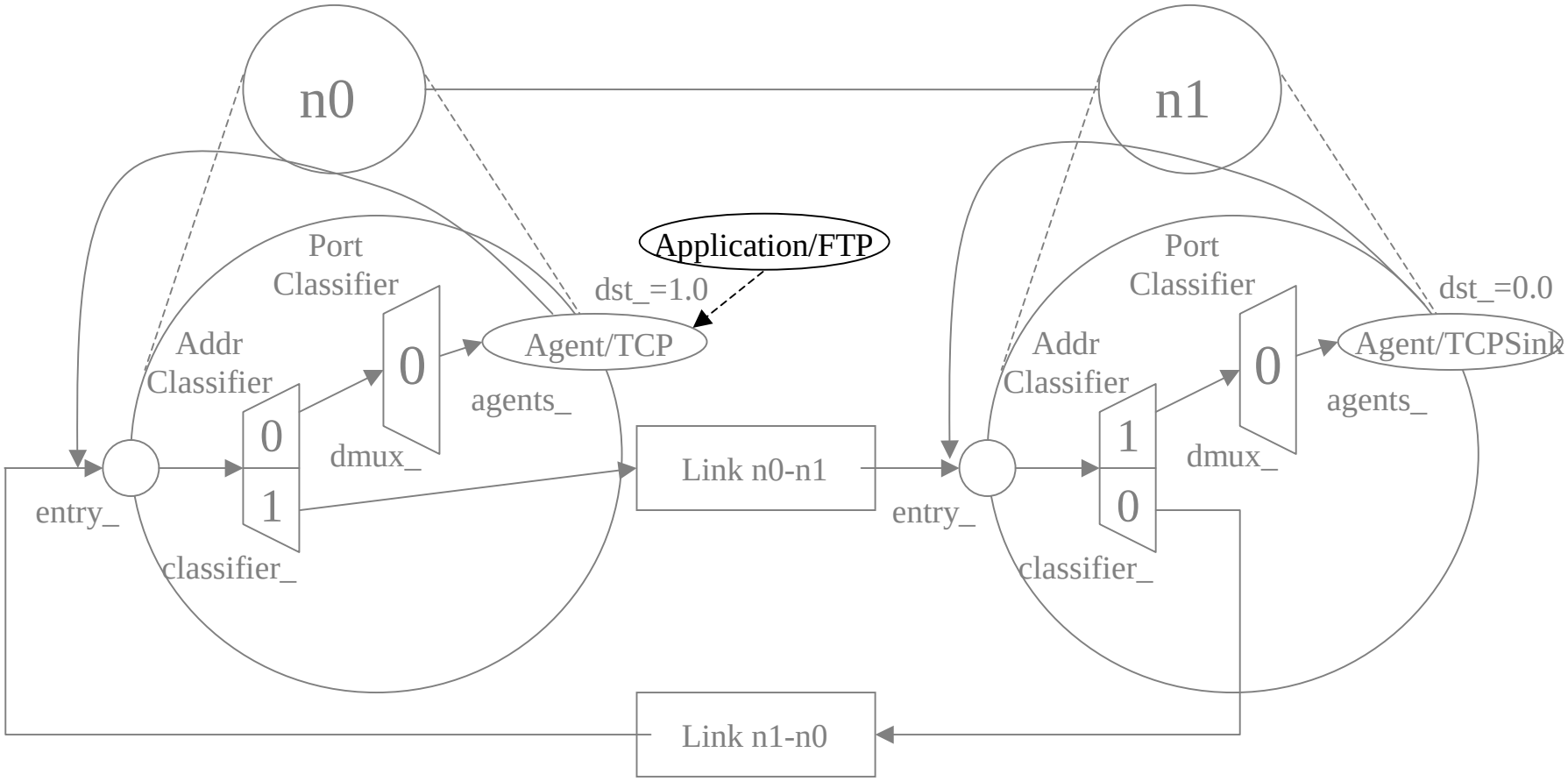
Routing (cont.)



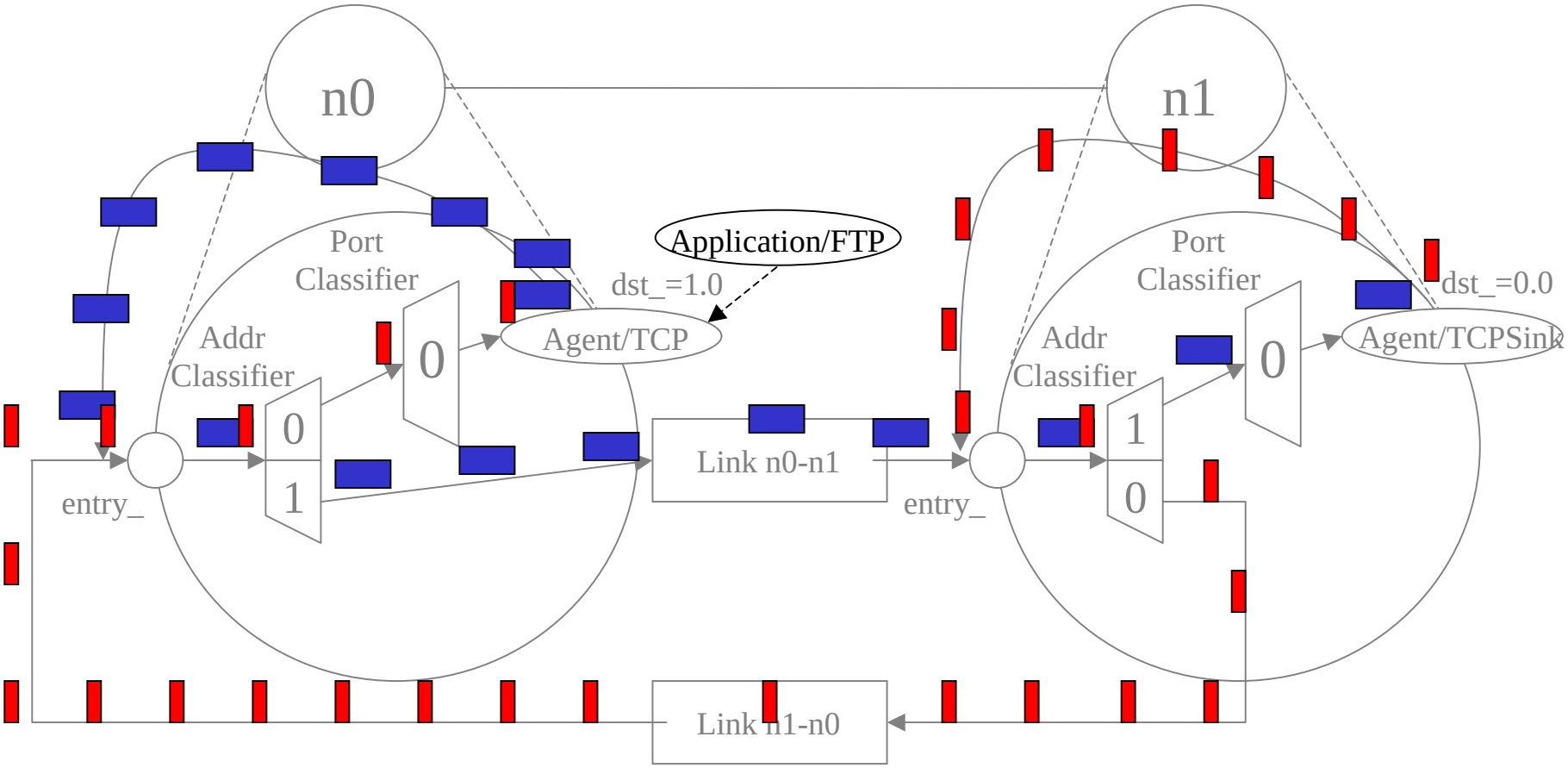
Transport


n0
n1
Port
Classifier
Addr
Classifier
0
1
dmux_
dst_=1.0
Agent/TCP
agents_
entry_
classifier_
Link n0-n1
Link n1-n0
dst_=0.0
Agent/TCPSink

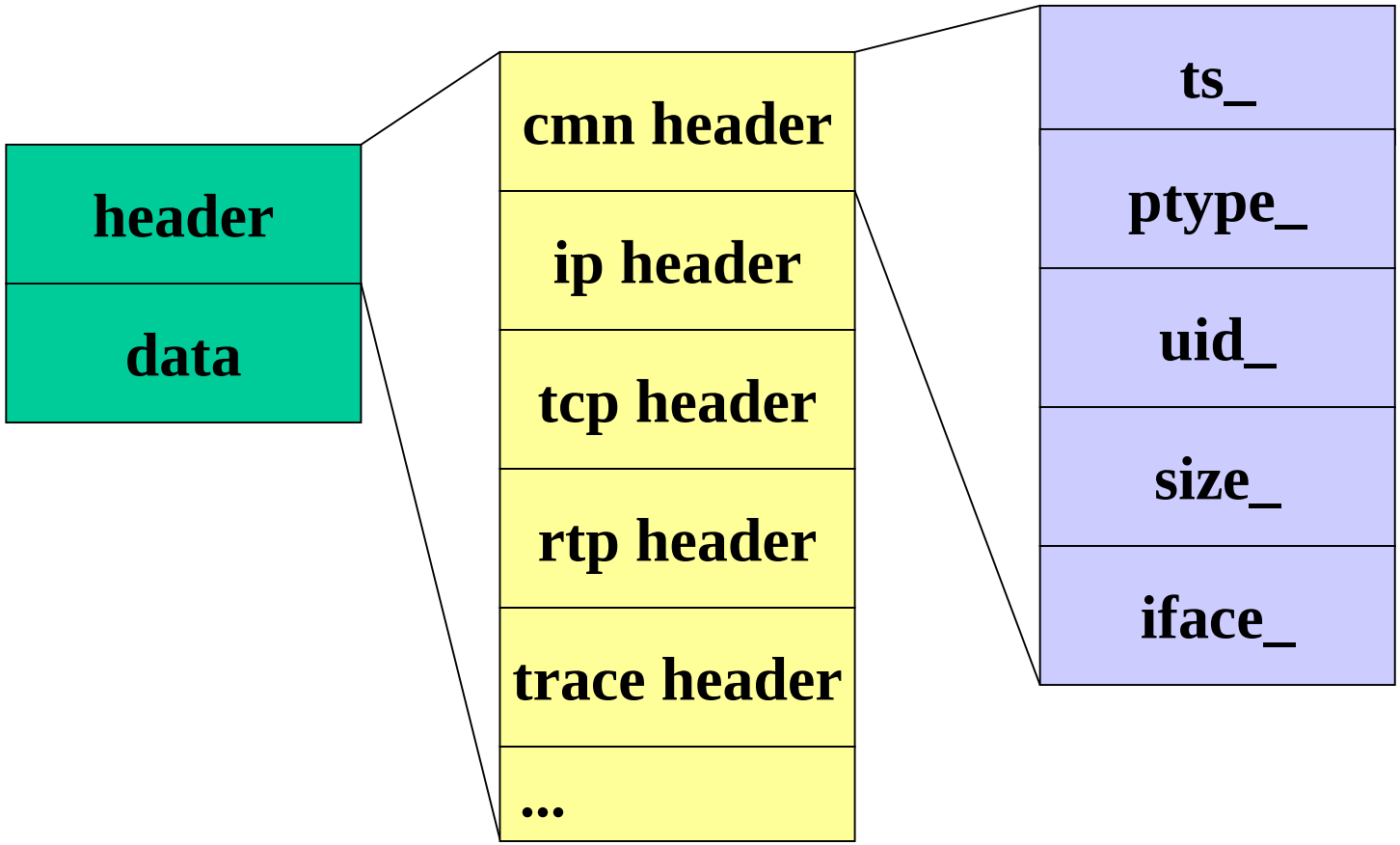
Application



Packet Flow


n0
n1
Application/FTP
Port
Classifier
dst_=1.0
Addr
Classifier
0
Agent/TCP
Port
Classifier
dst_=0.0
Addr
Classifier
0
Agent/TCPSink
0
1
entry_
Link n0-n1
entry_
1
0
Link n1-n0

Packet Format

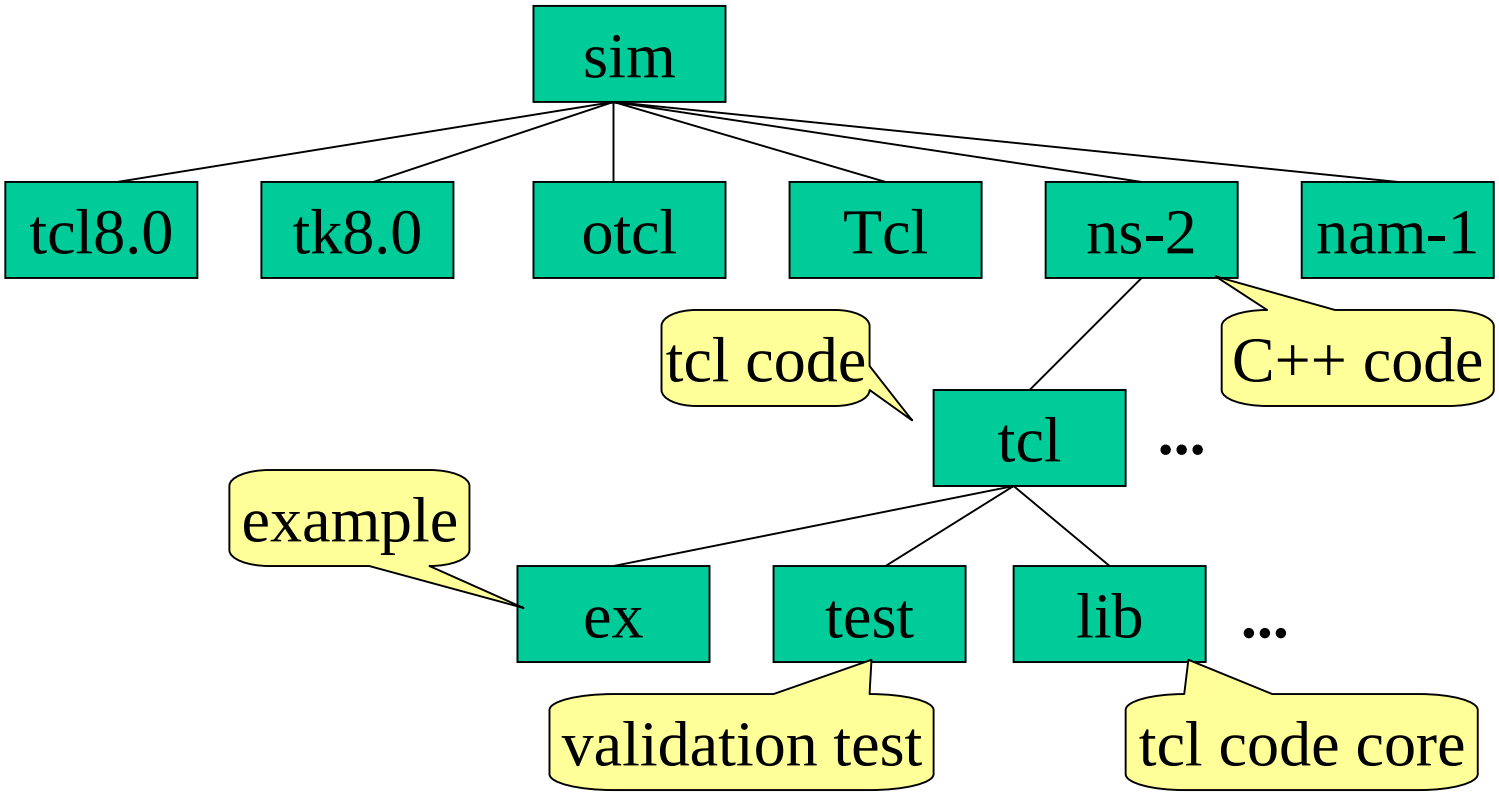


Extending ns-2 Simulator

Outline

- **Making changes**
- Creating new components

ns-2 Directory Structure



Making Changes in C++ Space

- Existing code
  - recompile
- Addition
  - change Makefile and recompile

Making Changes in otcl Space

- Existing code
  - recompile
  - source
- Addition
  - source
  - change Makefile (NS_TCL_LIB), tcl/ns-lib.tcl (source) and recompile

Outline

- Making changes
- **Creating new components**

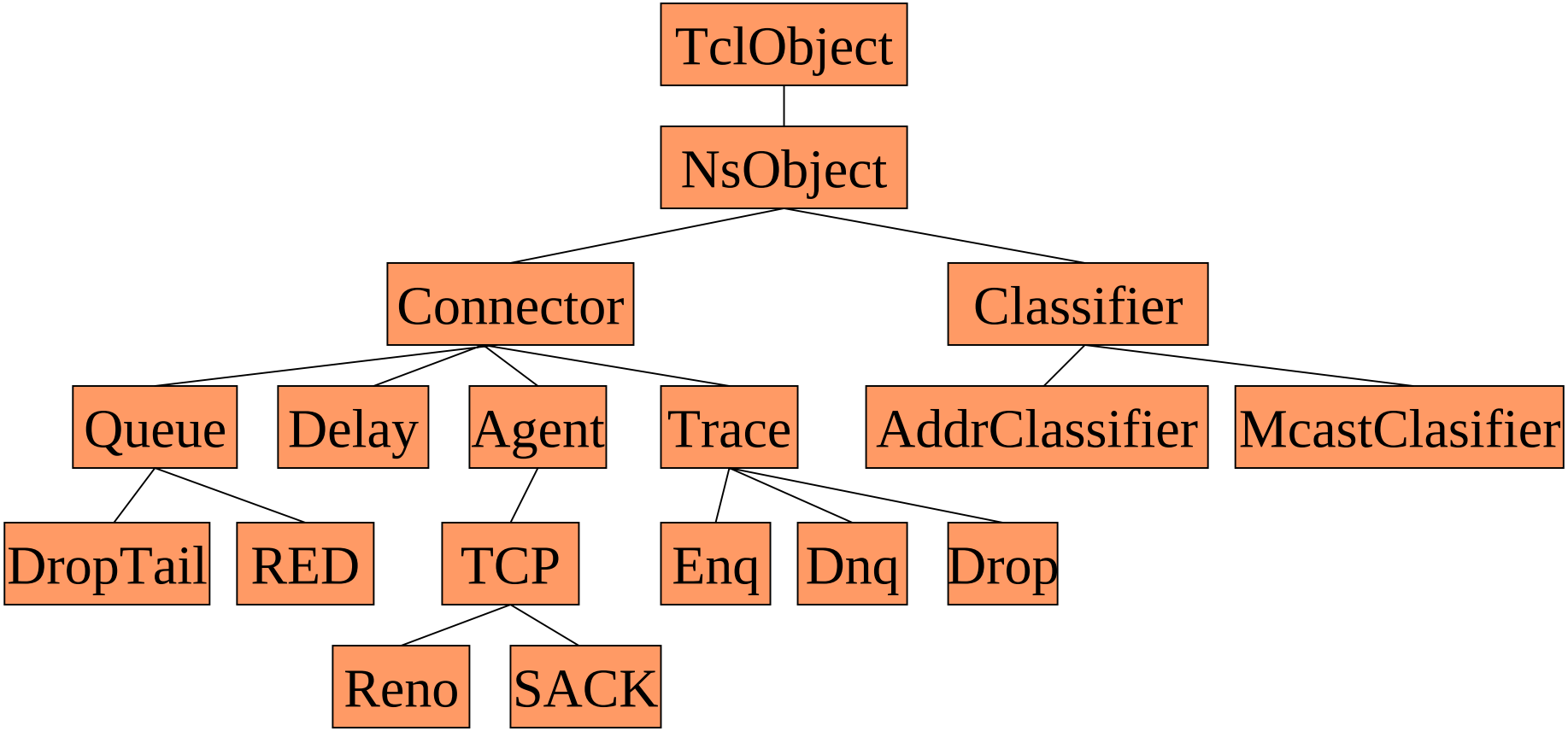
Creating New Components

- Guidelines
- Inheritance Hierarchy
- C++ and otcl Interface
- Examples
  - Network layer
  - Transport layer
  - Application layer

Guidelines

- Decide its inheritance structure
- Create the class and fill in the API virtual functions
- Define otcl linkage functions
- Write the necessary otcl code to access your agent

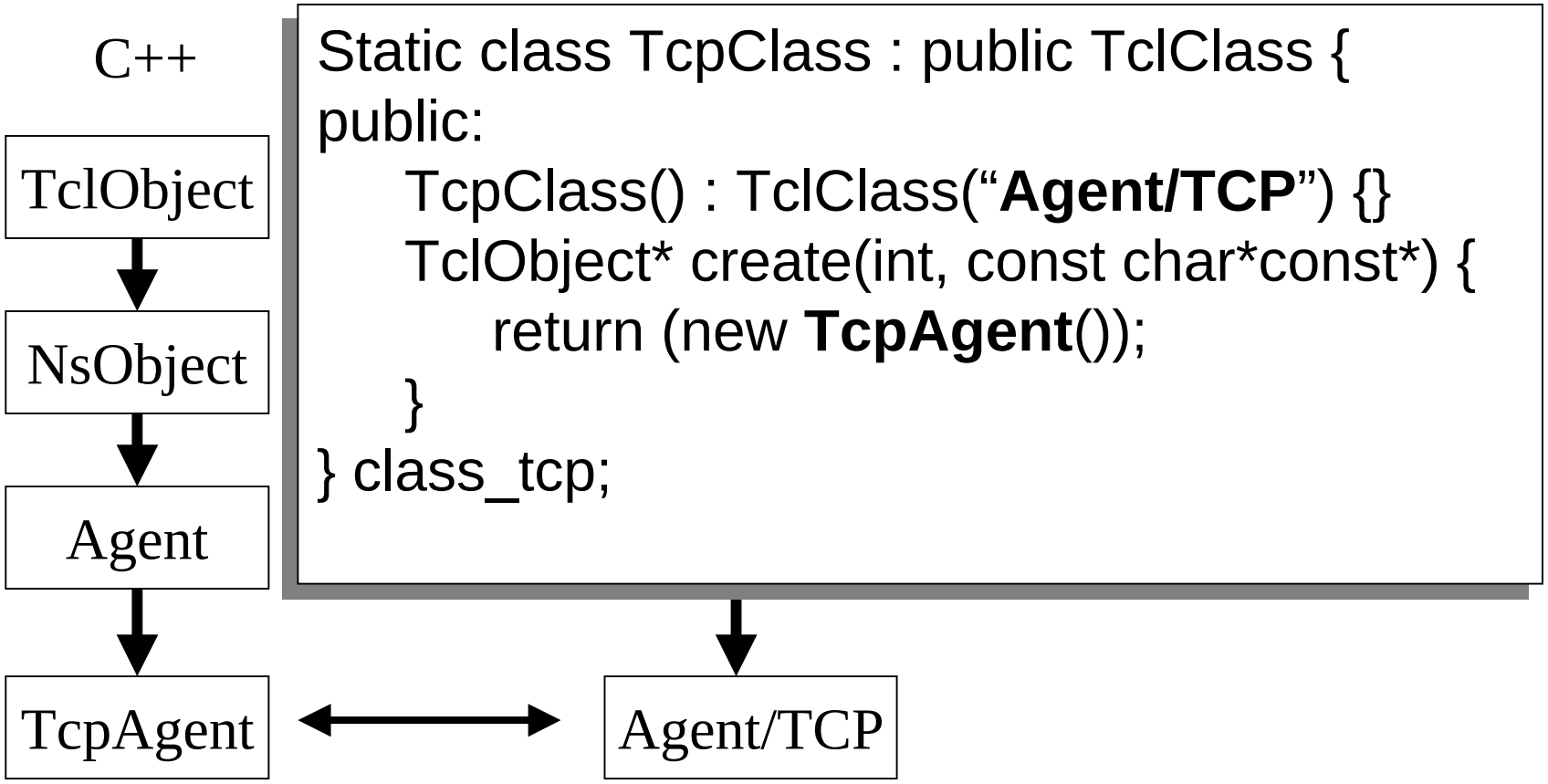
Class Hierarchy (Partial)



C++ and otcl Linkage

- TclClass
- TclObject: bind() method
- TclObject: command() method
- class Tcl

TclClass


C++
TclObject
NsObject
Agent
TcpAgent
Agent/TCP
Static class TcpClass : public TclClass {
public:
TcpClass() : TclClass("Agent/TCP") {}
TclObject* create(int, const char*const*) {
return (new TcpAgent());
}
} class_tcp;

TclObject: bind()

- C++

  TcpAgent::TcpAgent() {

  bind("**window**_", $wnd_);

  ...

  }

- otcl

  Agent/TCP set **window**_ 50

TclObject: command()

- C++

```
Int TcpAgent::command(int argc, const char*const* argv) {
        if (argc == 3) {
                if (strcomp(argv[1], “advance”) == 0) {
                        int newseq = atoi(argv[2]);
                        …
                        return(TCL_OK);
                }
        }
        return (Agent::command(argc, argv);
}
```

- otcl

```
set tcp [new Agent/TCP]
$tcp advance 10
```

Class Tcl

- C++

```
Tcl& tcl = Tcl::instance();
if (argc==2) {
        if (strcmp (argv[1], “now”) == 0) {
                tcl.resultf(“%g”, clock());
                return TCL_OK;
        }
        tcl.evalc(“puts hello, world”);
        tcl.error(“command not found”);
}
```

- otcl

```
foo now
foo whatever
```

Debugging

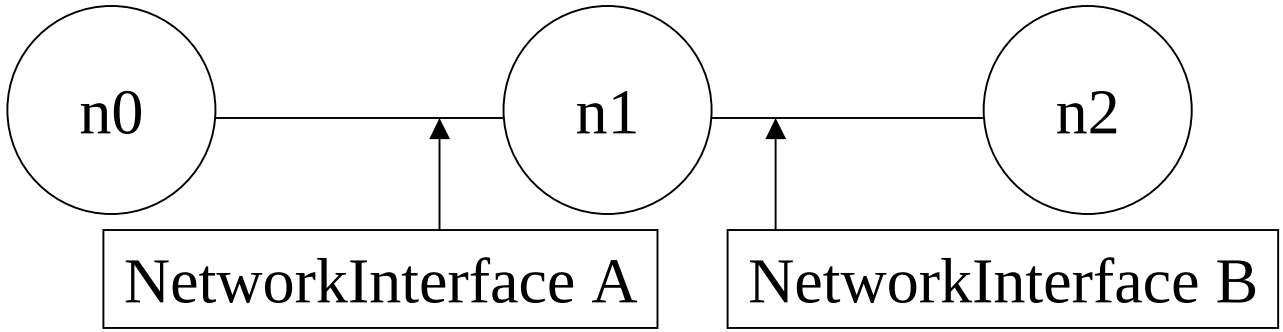
- printf() and puts “”
- gdb
- tcl debugger
  - http://expect.nist.gov/tcl-debug/
  - place `debug 1` at the appropriate location
  - trap to debugger from the script
  - single stepping through lines of codes
  - examine data and code using Tcl-ish commands

Case Studies

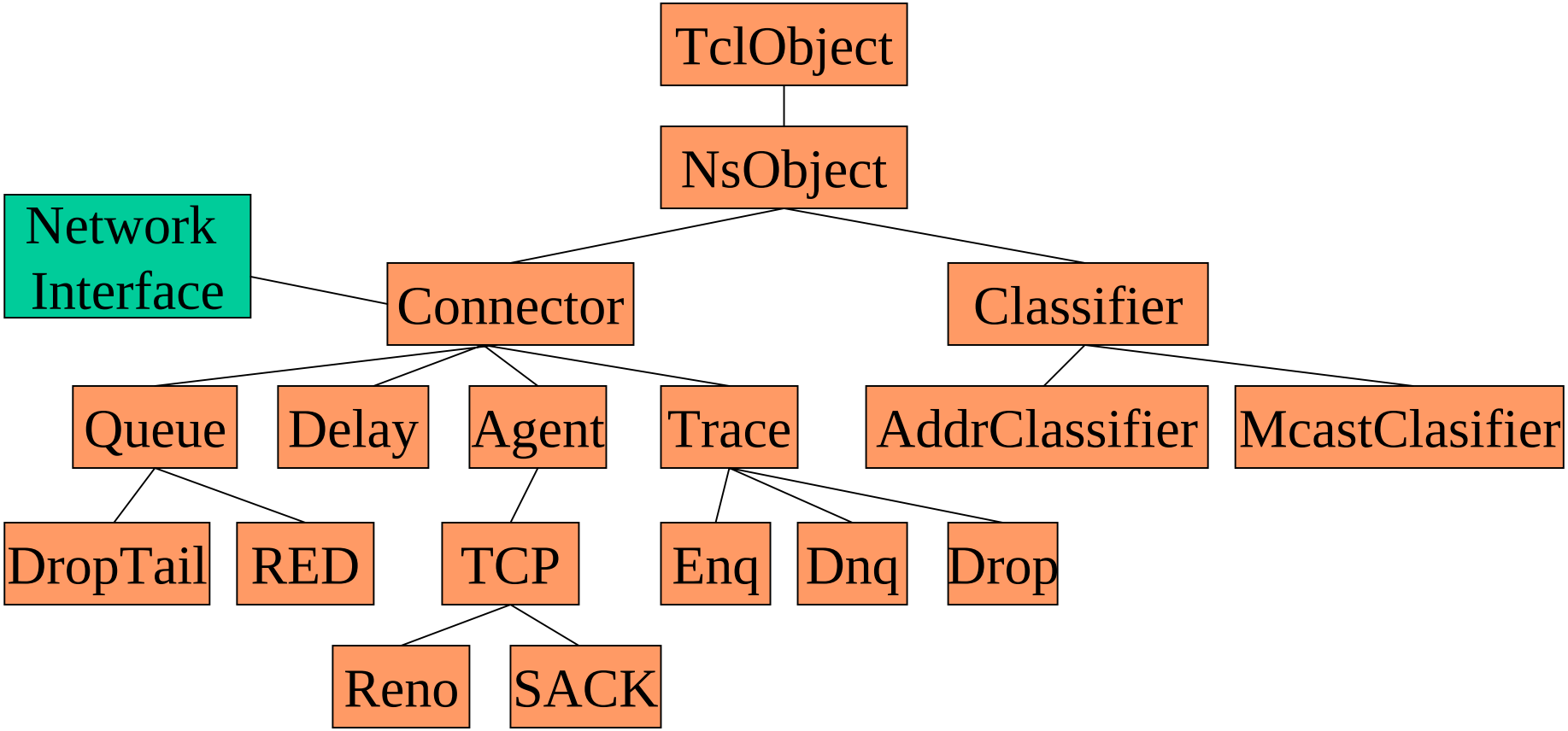
- Network Layer: Network Interface (packet labeler)
- Transport Layer: TCP Jump Start
- Application Layer: Message agent (ping)

Case 1: Network Layer

- Network Interface - Packet Labeler

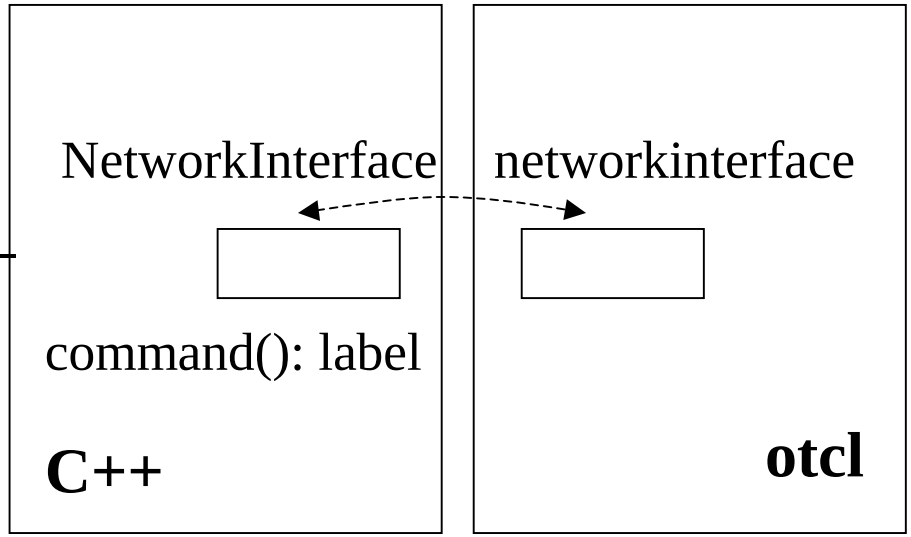


Class Hierarchy



Network Interface Labeler

- class NetworkInterface
  - NetworkInterface()
  - recv(pkt)
    - pkt->iface() = label_
  - send(pkt)
    - target->recv(pkt)
  - command()
    - label argv {label_ = argv}



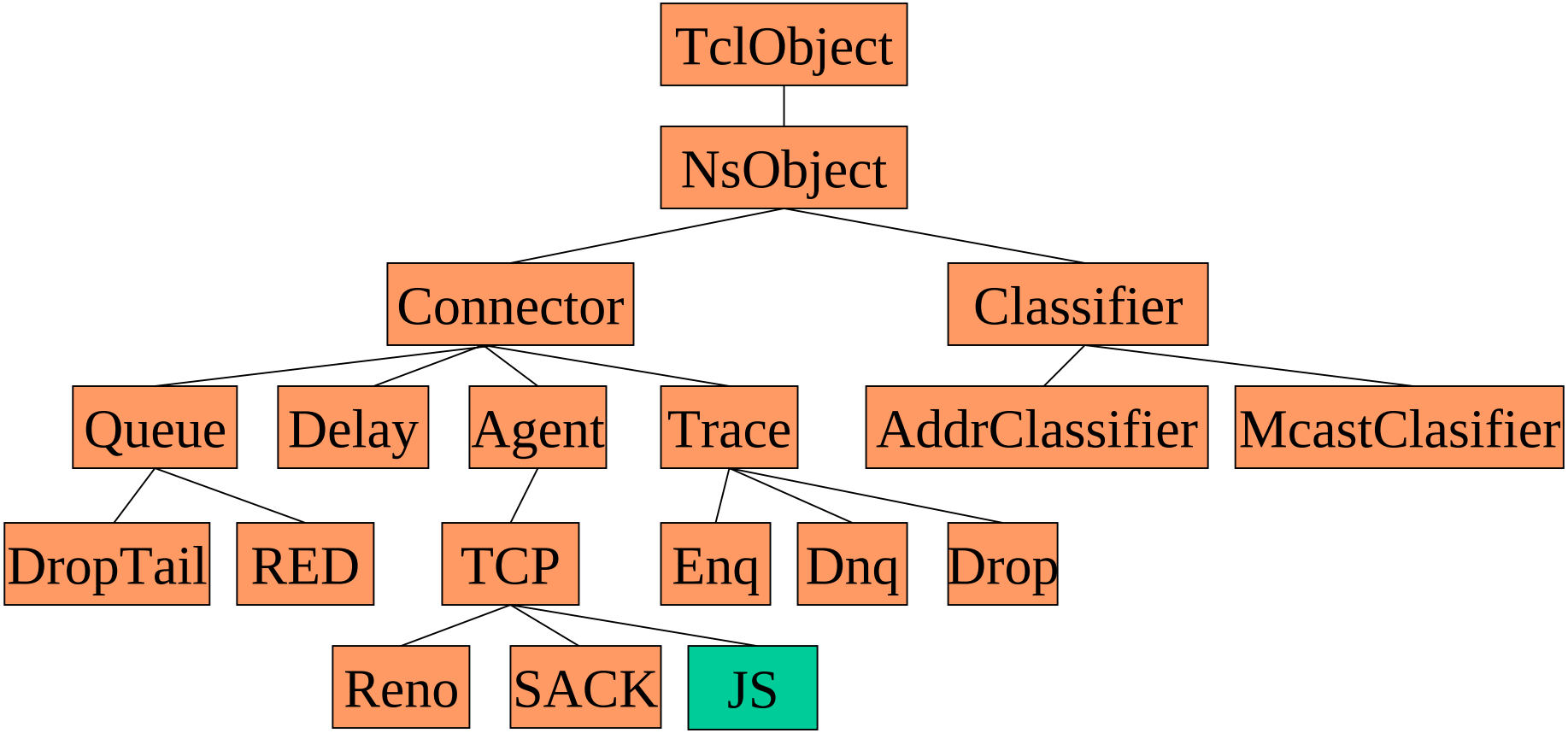
set iface [new networkinteface]
$iface label A

- TclClass("networkinterface")

Case 2: Transport Layer

- TCP Jump Start
  - from cwnd += 1
  - to cwnd = maxwin

Class Hierarchy

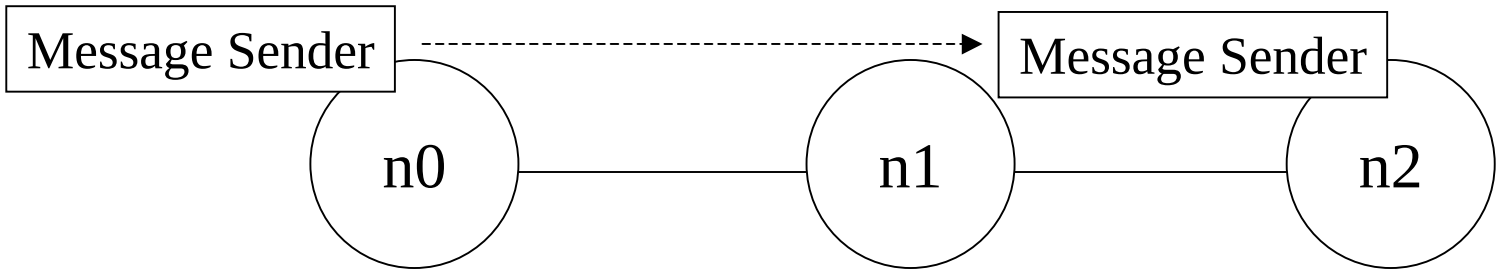


TCP Jump Start

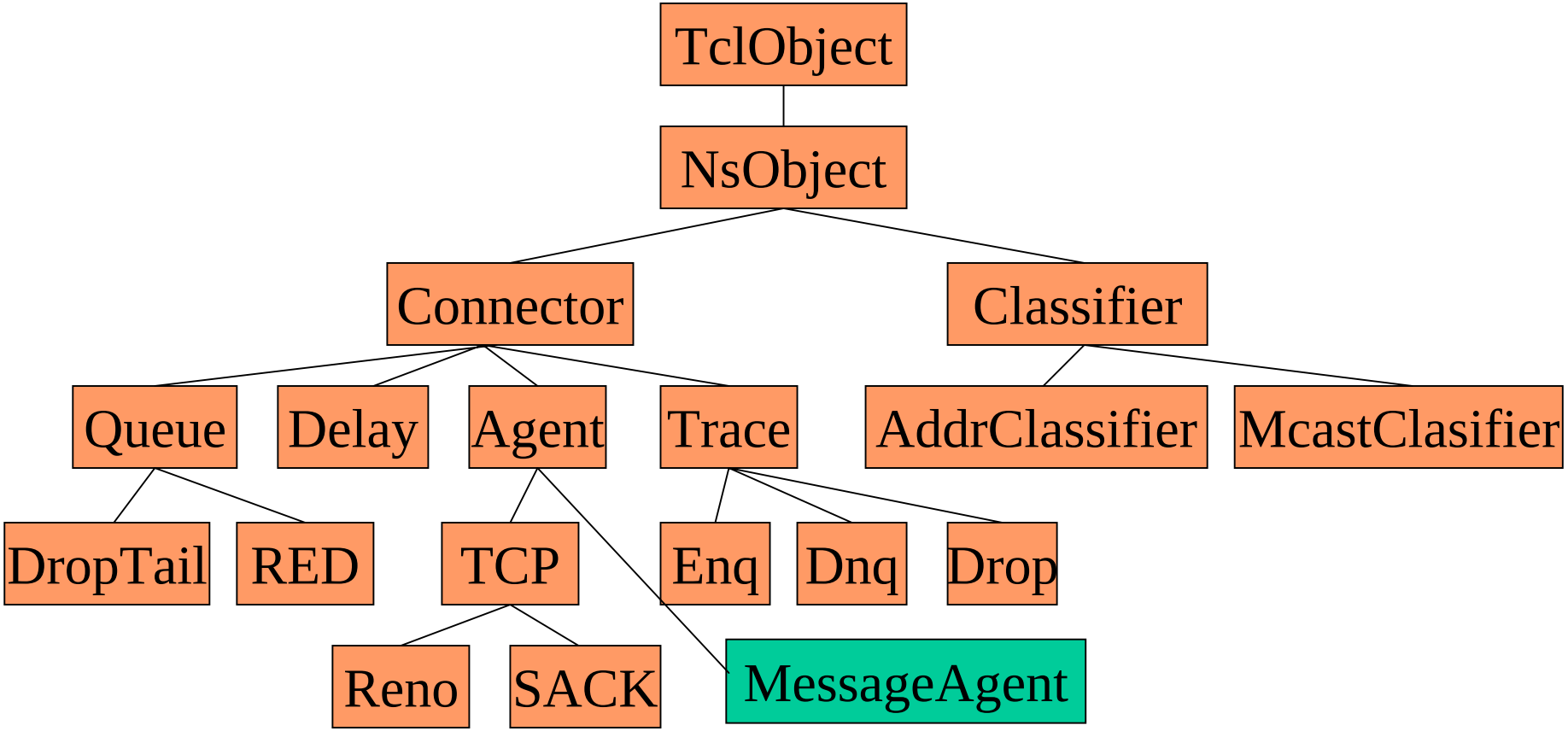
- class JSTcpAgent
  - openwin()
  - slowdown()
- TclClass("Agent/TCP/JS")

Case 3: Application Layer

- Message sender (ping)

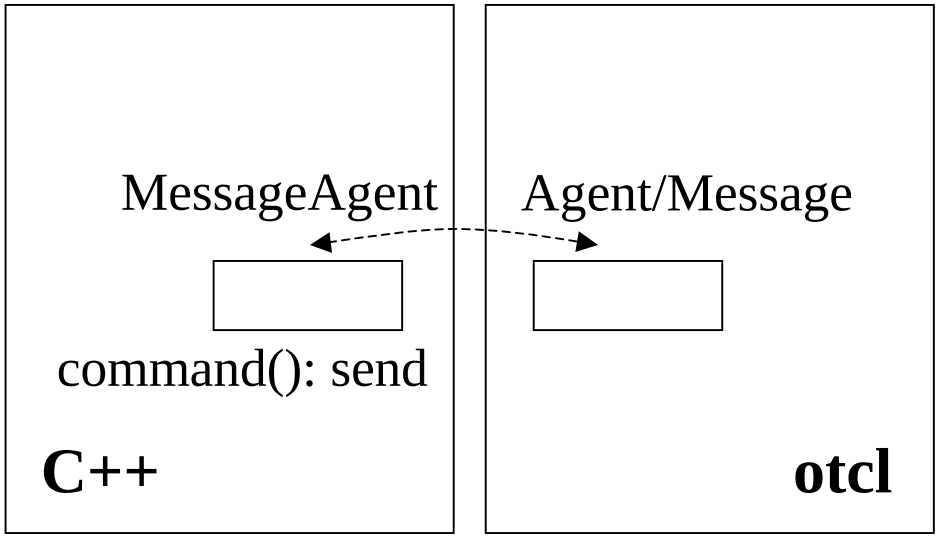


Class Hierarchy



Message Sender

- class MessageAgent
  - MessageAgent()
  - recv()
  - send()
  - command()
    - send {send()}



- TclClass("Agent/Message")

```
set msg [new Agent/Message]
$msg send
```

Special Topics

Special Topics

- Application Level Support
- Topology and Scenario Generation
- Wireless/Mobility Support

Application: Two-way TCP

- FullTcp connection
  - set tcp1 [new Agent/TCP/FullTcp]
  - set tcp2 [new Agent/TCP/FullTcp]
  - $ns attach-agent $n1 $tcp1
  - $ns attach-agent $n2 $tcp2
  - $ns connect $tcp1 $tcp2
  - $tcp2 listen

Application: TcpApp

- User data transfer
  - set app1 [new Application/TcpApp $tcp1]
  - set app2 [new Application/TcpApp $tcp2]
  - $app1 connect $app2
  - $ns at 1.0 “$app1 send <data_byte> \”<ns-2 command>\””
  - <ns-2 command>: will be executed when received at the receiver TcpApp

Topology Generation

- http://www-mash.cs.berkeley.edu/ns/ns-topogen.html

Packages	Graph Type	Edge Method
• ntg	• n-level	• prob
• GT-ITM	• flat, ts, n	• various
• TIERS	• 3-level	• spanning t
• rtg	• flat	• waxman

Scenario Generation

- http://www-mach.cs.berkeley.edu/ns/ns-scengeneration.html
- agent-gen-script.tcl
- Source generator files
  - source topo-gen.tcl
  - source agent-gen.tcl
  - source route-gen.tcl

topo-gen.tcl

- GT-ITM
- topology
  - outfile <file>
  - type <graph_type> : random or transit_stub
  - nodes <num_nodes>
  - connection_prob <probability>

route-gen.tcl

- Routing
  - outfile <file>
  - unicast <ucast_type>
  - multicast <mcast_type>

agent-gen.tcl

- Agents
  - outfile <file>
  - transport <transport_type>
  - src <application_type>
  - sink <transport_sink_type>
  - num <num_connections or %>
  - start <start_time>
  - stop <stop_time>

Wireless Support: Setup

- set ns [new Simulator]
- set chan [new Channel/WirelessChannel]
- set prop [new Propagation/TwoRayGround]
- set topo [new Topography]
- $topo load_flatgrid <length> <width>
- $prop topology $topo

Wireless Support: MobileNode

- Creating mobile nodes
  - set mnode [<routing>-create-mobile-node <node_id>]
  - <routing>: dsdv or dsr
- Node coordinates
  - $mnode set X_ <x>
  - $mnode set Y_ <y>
  - $mnode set Z_ 0

Mobility Support: Movement

- Specified
  - $ns at 1.0 “$mnode setdest <x> <y> <speed>”
- Random
  - $ns at 1.0 “$mnode start”